



# Soar IDE

Keith Knudsen, Mike Quist, Dave Ray & Bob Wray  
{knudsen,ray,quist} @soartech.com  
May 24, 2007

# Soar Editors and Development Environments

- A (very incomplete) list of development support tools for Soar ...

TAQL (5?) CMU, 1989	High-level language and toolset for Soar development	Operator templates
Soar Development Environment (6) UM, 1995	Emacs-based editor & debugger	Integrated editing and debugging; leveraged power of Emacs
TSI (7, 8) UM, 1998	Tcl/Tk-based debugging	Command macros; GUI-based commands
viSoar (7, 8?) Portsmouth, 1999	GUI-based editing environment	Early approach to datamap; explicit support for teamwork/STEAM
Visual Soar (8) UM, 2000	Full-featured editor * Prototype integration with Eclipse	Explicit support for ONC hierarchy idiom; datamap
HLBRL (8) PSU	High-level language and toolset for generation/creation of Soar programs	Explicit support for explanation,
Soar IDE (8) Soar Tech	Eclipse-based editor (Future debugging environment)	THIS TALK ☺

*"Building application domains creates a community with a large investment in ease of use, and hence with a willingness to expend the effort to make the tools to make [supporting and invigorating a theory] happen."*

[Newell, UTC]

# What is the Soar IDE?

The screenshot displays the Soar IDE interface within the Eclipse SDK. The main editor shows the source code for `elaborations.soar`, which includes a procedure `rhs-visual-range` and several stateful procedures (`sp`) for handling visual range and movement. A tooltip indicates a warning: "Variable <bogus> is not connected to state".

The Soar Datamap window shows a hierarchical view of the current state, including attributes like `air-hunter-coords`, `direction`, `dx`, `dy`, and `travel-range`.

The Problems window shows a list of errors, including a warning about the disconnected variable `<bogus>`.

**Yet another editor for the Soar language**

- Advanced parsing & error checking as you type
- Dynamically generated datamap as you type
- Support for embedded TCL

• Plugin to Eclipse development environment

• Public beta release this fall!

# Why did we build it?

- Unsatisfied with constraints imposed by existing options
  - UofM operator style limitations
  - SoarTech heavily invested in inline Tcl
- Gained Eclipse experience after developing several other plugins
- Believed that building a more powerful (yet flexible) Tcl/Soar parser into the editor would:
  - Catch errors earlier
  - Reduce debugging time
  - Improve Soar code understandability
  - --> Improve productivity

# Why Eclipse?



- Well-supported, stable, extensible environment
- Cross-platform support
- Consistent interface for multiple plugins
- Can include Soar files in a project with files of other types (Java, HLSR, XML, C++, HTML ...)
- Plugin support for version control (Subversion, CVS, etc.)
- Significant base functionality to build on
  - Project organization
  - Flexible UI
  - Customizable syntax highlighting
  - Code expansion templates
  - Error & warning reporting interface
  - Update manager
  - Regex cross file search and replace
  - Diff tool integrated with version control & local changes history
  - Advanced help tools (traditional & pop-up help)

# What we built: 1) Soar Perspective

The screenshot displays the Eclipse IDE in the Soar Perspective. The main editor shows Soar code with a variable error: "Variable <bogus> is not connected to state". The Package Explorer on the left shows a project structure for "AirHunters". The Soar Datamap at the bottom left shows a table of variables and their values. The Outline view on the right lists Soar productions and procedures. The Problems view at the bottom shows the error message.

Attribute	Values
air-hunter-coords (?)	
after-move (?+)	
command (+)	move
direction (+)	
dx (?+)	
dy (?+)	
travel-range (+)	
visual-range (+)	
command (+)	wait
direction (+)	
dx (+)	

- Java Package Explorer
  - Browse all files in the project; see iconic indications of errors, warnings, to-do's, file and directory types
  - Compare files to each other, to Subversion repository, or to local histories
- Soar Explorer
  - Browse Soar files or productions in project
  - Iconic indications of errors, warnings, etc
- Outline View
  - Contents of active editor
    - Productions
    - Tcl procedures and variables
    - Import ("source") statements
- Problem reporting
- Full-featured code editor, Soar Source Viewer & Datamap (continued...)

# What we built: 2) Soar Editor

```
# visual range is in metric where each square is d=1 from its e
# this distance is the larger of the two absolute values |dx| e
proc rhs-visual-range { dx dy } {
    return "(max (abs $dx) (abs $dy))"
}

sp "hunter-coords*elaborate*after-move*visual-range
(state <s> ^<< air-hunter-coords ground-hunter-coords >> <h-
(<h-coords> ^after-move <am>)
<am> ^dx <dx>
^dy <dy>)

-->
(<am> ^visual-range [rhs-visual-range <dx> <dy>])
"

sp {hunter-coords*elaborate*after-move*zero-displacement
(state <s> ^<< air-hunter-coords ground-hunter-coords >> <h-
^directions.value <dir>)

-->
(<bogus> ^q 5)
(<h-coords> ^after-move <h-coords>)
(<h-coords> ^command wait
^direction <dir>)
}
```

Attribute	Values
air-hunter-coords (?)	
after-move (?)	
command (+)	move
direction (+)	
dx (?)	
dy (?)	
travel-range (+)	
visual-range (+)	
command (+)	wait
direction (+)	
dx (+)	

- Syntax checking
  - Highlights syntax errors while you type
  - Errors collected in Problems View
  - Catches some non-syntactic errors as well (scoping, etc.)
  - Syntax coloring
- Line-by-line edit histories
- “Hover help” for Soar and Tcl commands
- Soar templates (sp) and keywords (excise, waitsnc, ...)
- Tcl expansion (see slide)
- Dynamic datamap (see slide)





# What we built: 4) Soar Source Viewer

The screenshot displays the Eclipse IDE interface for the Soar Source Viewer. The Package Explorer on the left shows a project structure with 'elaborations.soar' selected. The Soar Explorer shows expanded Tcl code for 'visual-range'. The Soar Datamap shows a table of attributes and values. The Problems window shows a warning about a variable not connected to state.

Attribute	Values
air-hunter-coords (?)	
after-move (?)	move
command (+)	
direction (+)	
dx (?)	
dy (?)	
travel-range (+)	
visual-range (+)	
command (+)	wait
direction (+)	
dx (+)	

```
# visual range is in metric where ea
# this distance is the larger of the
proc rhs-visual-range { dx dy } {
    return "(max (abs $dx) (abs $dy))"
}

sp "hunter-coords*elaborate*after-move
(state <s> ^<< air-hunter-coords
(<h-coords> ^after-move <am>)
(<am> ^dx <dx>
 ^dy <dy>)

-->
(<am> ^visual-range [rhs-visual-range <dx> <dy>])
"

sp {hunter-coords*elaborate*after-move*zero-displacement
(state <s> ^<< air-hunter-coords ground-hunter-coords >> <h-
^directions.value <dir>)

-->
(<bogus> ^q 5)
(<h-coords> ^after-move <h-coords>)
(<h-coords> ^command wait
 ^direction <dir>)
}
```

Variable <bogus> is not connected to state

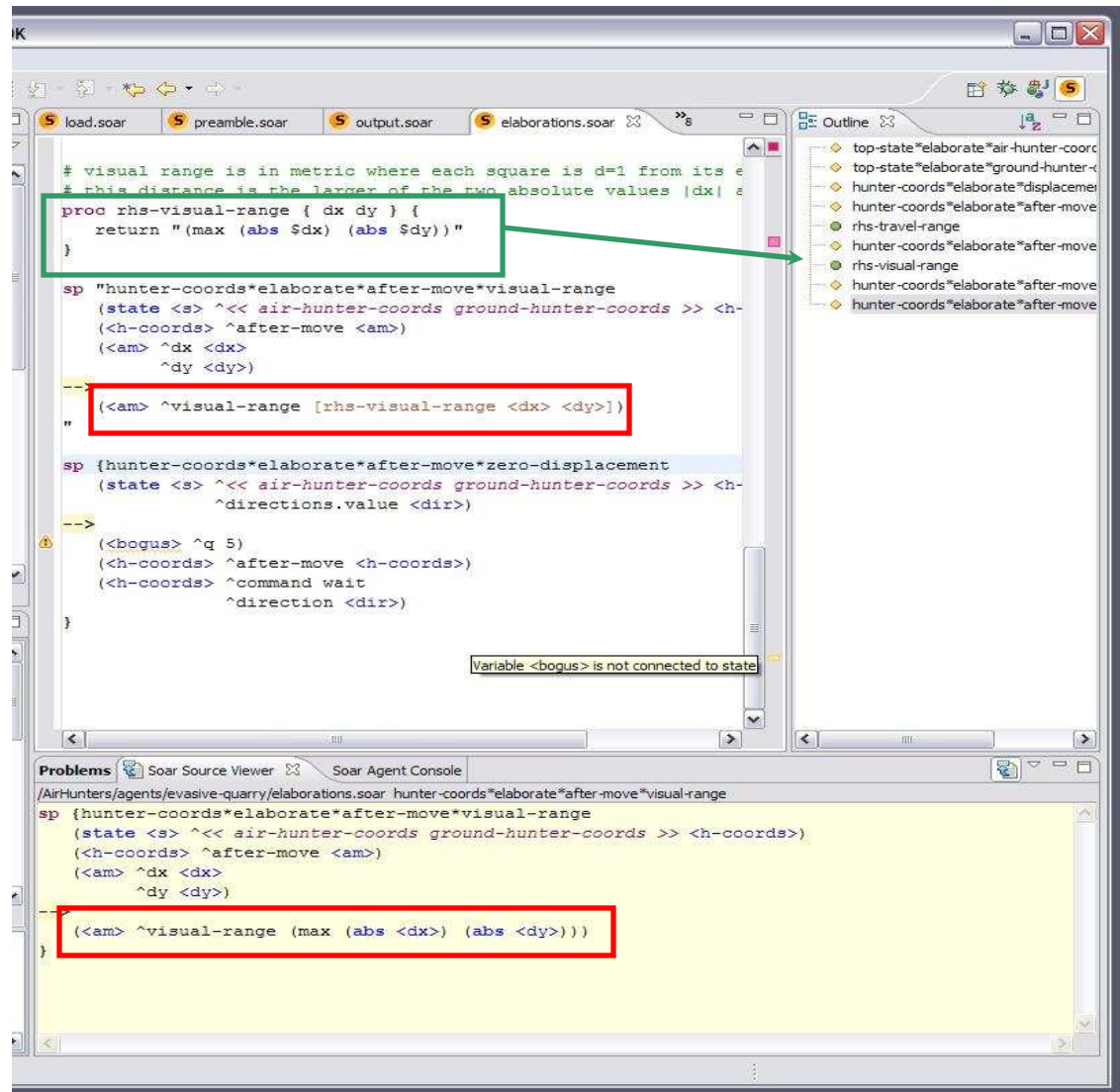
```
/AirHunters/agents/evasive-quarry/elaborations.soar hunter-coords*elaborate*after-move*visual-range
sp {hunter-coords*elaborate*after-move*visual-range
(state <s> ^<< air-hunter-coords ground-hunter-coords >> <h-coords>)
(<h-coords> ^after-move <am>)
(<am> ^dx <dx>
 ^dy <dy>)

-->
(<am> ^visual-range (max (abs <dx>) (abs <dy>)))
}
```

- Displays source code of current selection of active view (Package Explorer, Soar Explorer, Editor, etc)
  - Soar files
  - Productions
  - Tcl Procedures
- Shows expanded Tcl (see slide)
- Useful for quickly browsing without opening several editors

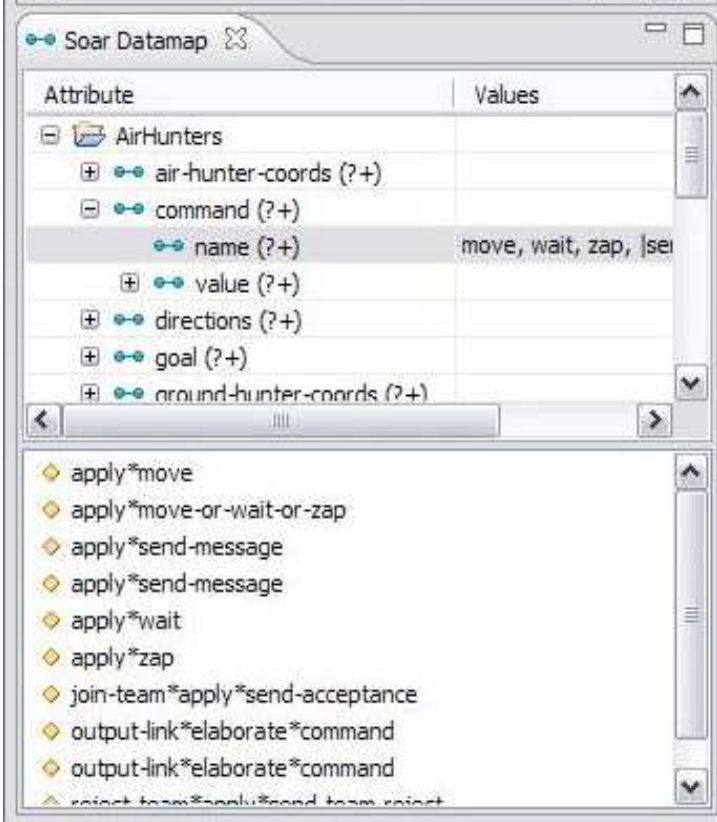
# What we built: 5) Tcl Expansion

- View original and Tcl-expanded source simultaneously
- Live updating of Tcl macro and variable definitions
- Tcl procedures appear in outline view



## What we built: 6) Dynamic Datamap

- Shows tests (?), assignments (+), and values of attributes
  - Context-sensitive, e.g., operator and goal 'name' attributes are distinct
  - Hierarchy reflects the structure of working memory
- Updates along with code changes
- Linked to original productions
  - Easily locate code that reads individual portions of input link, or that writes specific output-link commands



The screenshot shows a window titled "Soar Datamap" with a table of attributes and their values. The table has two columns: "Attribute" and "Values". The attributes are organized into a hierarchy under "AirHunters".

Attribute	Values
AirHunters	
air-hunter-coords (?+)	
command (?+)	
name (?+)	move, wait, zap,  ser
value (?+)	
directions (?+)	
goal (?+)	
ground-hunter-coords (?+)	

Below the table, there is a list of productions (rules) with diamond-shaped icons next to them:

- apply\*move
- apply\*move-or-wait-or-zap
- apply\*send-message
- apply\*send-message
- apply\*wait
- apply\*zap
- join-team\*apply\*send-acceptance
- output-link\*elaborate\*command
- output-link\*elaborate\*command
- reject-team\*apply\*send-team-reject

# How is Soar IDE different from Visual Soar?

- Soar IDE benefits from being an Eclipse plugin
  - Increasingly seems to be a significant advantage
  - 100s of features in Eclipse base
  - 1000s of features easily adopted from other Eclipse language plugins
- In-place TCL code expansion
  - Killer feature from SoarTech's perspective, but does anyone else use TCL for Soar anymore?
- No constraints on directory structure
  - Really useful for pulling in legacy code
- No constraints on UofM operator style
  - But also no benefits
- Dynamic datamap generated as you type
  - But...not partitioned by operator/problem space
  - Just a big representation of working memory. Still useful, but may be extended in the future to handle filtering by operator

# Technical Hurdles

- Tcl parsing
  - Tcl is an extremely flexible language, difficult to find errors with just a basic parse.
  - A random block of Java code is probably valid Tcl :)
- Tcl expansion on the fly and how to present it to the user
  - Previous experience with integrating Tcl with SML made this less painful.
- Creating a parser that gives useful error information
  - File/line/column style error reports are ok for command-line tools
  - Eclipse works much better when given character ranges. Allows for nice underlying of bugs.
- Significant modifications to Visual Soar parser
  - Errors reported as ranges rather than line/column
  - Parse production bodies individually. Essential for parsing results of Tcl expansion.
- Performance
  - Cache parse information and other metadata (necessary for large projects like TacAirSoar)

# Eclipse Lessons Learned

- Use Eclipse to its fullest extent
  - The Soar IDE code is highly coupled to Eclipse APIs.
  - This made a huge difference in ease of implementation.
- Study of JDT source code for design
  - We found that studying the Eclipse source code is essential for really making quality plugins.
  - Books and online articles often only scratch the surface.
- Other benefits of studying Eclipse source
  - Doing things the "Eclipse Way" leads to better integration with other plugins
  - Avoids reinventing the wheel.

# First Impressions from Soar Programmers

- Eclipse learning curve is steep
  - ... Because the environment is so rich. File histories, integrated CVS/Subversion access, visual diffs, maintenance of warnings and to-do's, make it worth learning. And many people use Eclipse already.
- Soar IDE editing features are *outstanding*
  - Especially useful for larger projects,
  - Tcl-heavy projects, Integration projects (e.g., Java & Soar), and
  - projects with multiple developers (due to browsing features and integration w/ version control)
- Just starting to benefit from Dynamic Datamap
  - Soar developers really like it
  - Still learning how to best take advantage of it
  - Rich source of feature requests
- Could use tighter integration with Java Soar Debugger
  - Doug?

# Demonstration

- Java TankSoar demo
  - Included in the latest Soar distributions
  - Includes Java and Soar code
  - Will demonstrate the views and features just mentioned.
- How to get the IDE?
  - Public **beta** release
    - Release planned this fall
    - Currently being tested internally
  - <http://www.soartech.com/downloads.soar-ide.php>
  - soar-ide@soartech.com
    - <http://webmail.soartech.com/mailman/listinfo/soar-ide>
  - {knudsen, ray, quist} @soartech.com